GPU Acceleration of Communication Avoiding Chebyshev Basis Conjugate Gradient Solver for Multiphase CFD Simulations

Yussuf Ali, Naoyuki Onodera, **Yasuhiro Idomura**,
Center for Computational Science and e-Systems, Japan Atomic Energy Agency

Takuya Ina, T. Imamura
RIKEN Center for Computational Science

10th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems
November 18, 2019, Denver, CO, USA

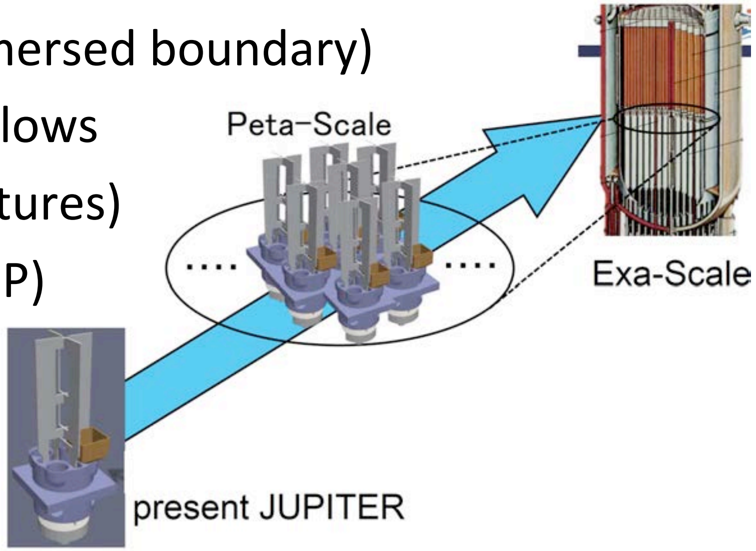
Acknowledgements

- S. Yamashita, S. Yamada, Y. Hasegawa (JAEA)
- This work is supported by MEXT (Grant for Post-K priority issue 6), OLCF (DD project), JCAHPC (Oakforest-PACS grand challenge), HPCI(hp190073), JHPCN (jh190050) .
- Computation is performed on Oakforest-PACS@JCAHPC, Reedbush@U.Tokyo, Tsubame3.0@Tokyo Tech., ABCI@AIST, SUMMIT@ORNL, and ICEX@JAEA.

Exa-scale simulations for severe accident analysis

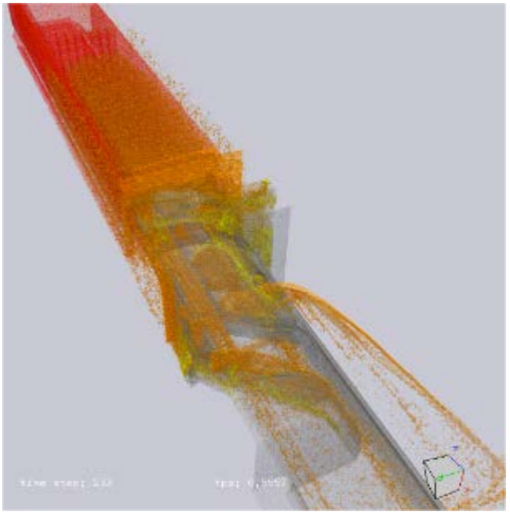
- JAEA promotes the development of multiphase thermal-hydraulic CFD code for analyzing severe accidents in the Fukushima Daiichi Nuclear Power Plant

- JUPITER code [Yamashita,NED2017] simulates relocation of molten materials in nuclear reactors as incompressible viscous fluids.
  - Finite difference in structured grids (immersed boundary)
  - Volume of fluid method for multiphase flows
  - Multi-components (fuel, absorber, structures)
  - 3D domain decomposition (MPI+OpenMP)



- Target problems
  - Peta-scale (K-computer, Tsubame3.0)
    - Simulate melt-relocation behavior of several fuel assemblies
  - Exa-scale
    - Severe accident analysis for whole reactor pressure vessel

Pressure Poisson solver in JUPITER



- Melt relocation of fuel assemblies
  - Solid/Liquid phases of UO_2, Zry, B_4C, SUS, and Air
  - Problem size: 1,280x1,280x4,608~7.5G grids
- Pressure Poisson Solver

$$\nabla \cdot \mathbf{u}^{n+1} = \nabla \cdot \mathbf{u}^{*} - \nabla \cdot \left(\frac{\Delta t}{\rho} \nabla p \right) = 0$$

  - Pressure Poisson solver occupies more than 90% of the total cost
  - 2^{nd} order centered finite difference in structured grids (7-stencils)
  - Large density contrast ~10^7 of multiphase flows gives an ill-conditioned problem, and its condition becomes worse in larger problems

    →Preconditioner is essential
  - Communication Avoiding (CA) Krylov solvers on CPU platforms

    [A. Mayumi, Y. Idomura, T. Ina, et al., Proc. ScalA'16@SC16 (2016)]

    [Y. Idomura, T. Ina, A. Mayumi, et al., Lecture Notes Comput. Science 10776, 257 (2018)]

    [Y. Idomura, T. Ina, S. Yamashita, et al., Proc. ScalA'18@SC18 (2018)]

    →In this work, we develop CA-Krylov solvers on GPU platforms

Krylov solvers for Pressure Poisson equation

A: symmetric block diagonal matrix

Algorithm Preconditioned Conjugate Gradient method

Require: $A\mathbf{x} = \mathbf{b}$, Initial guess $\mathbf{x}_1$

$\mathbf{r}_1 := \mathbf{b} - A\mathbf{x}_1, \mathbf{z}_1 = M^{-1}\mathbf{r}_1, \mathbf{p}_1 := \mathbf{z}_1$
for $j = 1, 2, \ldots$ until convergence do
  Compute $\mathbf{w} := A\mathbf{p}_j$
  $\alpha_j := \langle \mathbf{r}_j, \mathbf{z}_j \rangle / \langle \mathbf{w}, \mathbf{p}_j \rangle$
  $\mathbf{x}_{j+1} := \mathbf{x}_j + \alpha_j \mathbf{p}_j$
  $\mathbf{r}_{j+1} := \mathbf{r}_j - \alpha_j \mathbf{w}$
  $\mathbf{z}_{j+1} := M^{-1}\mathbf{r}_{j+1}$
  $\beta_j := \langle \mathbf{r}_{j+1}, \mathbf{z}_{j+1} \rangle / \langle \mathbf{r}_j, \mathbf{z}_j \rangle$
  $\mathbf{p}_{j+1} := \mathbf{z}_{j+1} + \beta_j \mathbf{p}_j$
end for

SpMV
Precon
AXPY

Chebyshev Basis Communication-Avoiding CG [Suda,RISM2016]

Algorithm Chebyshev Basis CACG (P-CBCG) method

Require: $A\mathbf{x} = \mathbf{b}$, Initial guess $\mathbf{x}_0$

$\mathbf{r}_0 := \mathbf{b} - A\mathbf{x}_0$
Compute S_0 $(T_0(AM^{-1})\mathbf{r}_0, \ldots, T_{s-1}(AM^{-1})\mathbf{r}_0)$
$Q_0 = S_0$
for $k = 0, 1, 2, \ldots$ until convergence do
  Compute $Q_k^* A Q_k$
  Compute $Q_k^* \mathbf{r}_{sk}$
  $\mathbf{a}_k := (Q_k^* A Q_k)^{-1} Q_k^* \mathbf{r}_{sk}$
  $\mathbf{x}_{s(k+1)} := \mathbf{x}_{sk} + Q_k \mathbf{a}_k$
  $\mathbf{r}_{s(k+1)} := \mathbf{r}_{sk} - AQ_k \mathbf{a}_k$
  Compute
  S_{k+1} $(T_0(AM^{-1})\mathbf{r}_{s(k+1)}, \ldots, T_{s-1}(AM^{-1})\mathbf{r}_{s(k+1)})$
  Compute $Q_k^* A S_{k+1}$
  $B_k := (Q_k^* A Q_k)^{-1} Q_k^* A S_{k+1}$
  $Q_{k+1} := S_{k+1} - Q_k B_k$
  $AQ_{k+1} := AS_{k+1} + AQ_k B_k$
end for

SpMV+Precon
GEMV
GEMM

- Comparisons of P-CG and P-CBCG (s=12) [Idomura,LNCS2018]

	P-CG	P-CBCG	P-CBCG/PCG
All_reduce/iteration	2	2/s	1/s
Computation [Flop/grid]	39.0	123.7	3.17
Memory access [Byte/grid]	248.0	312.0	1.26
Roofline time on ICEX [ns/grid]	4.33	5.61	1.30
Elapse time on ICEX [ns/grid]	5.19	6.71	1.30

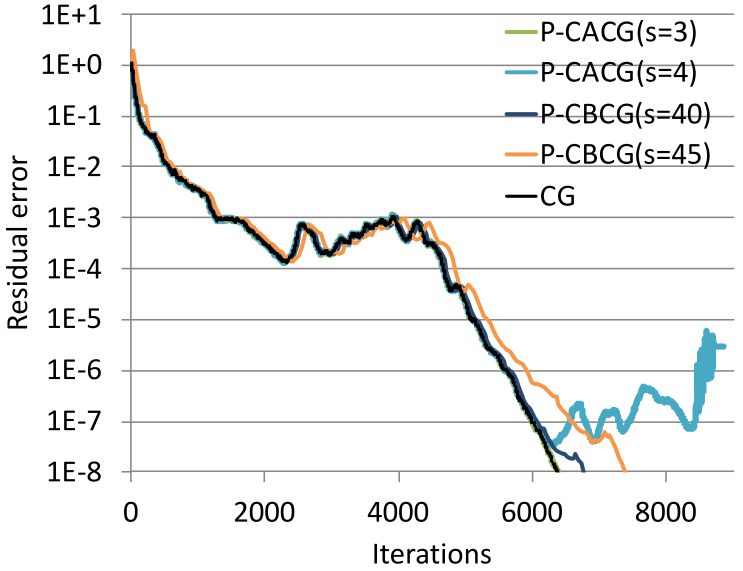
- ICEX@JAEA: Xeon E5-2680v3 (Haswell), B/F=0.12
- Roofline model [Shimokawabe,SC10]

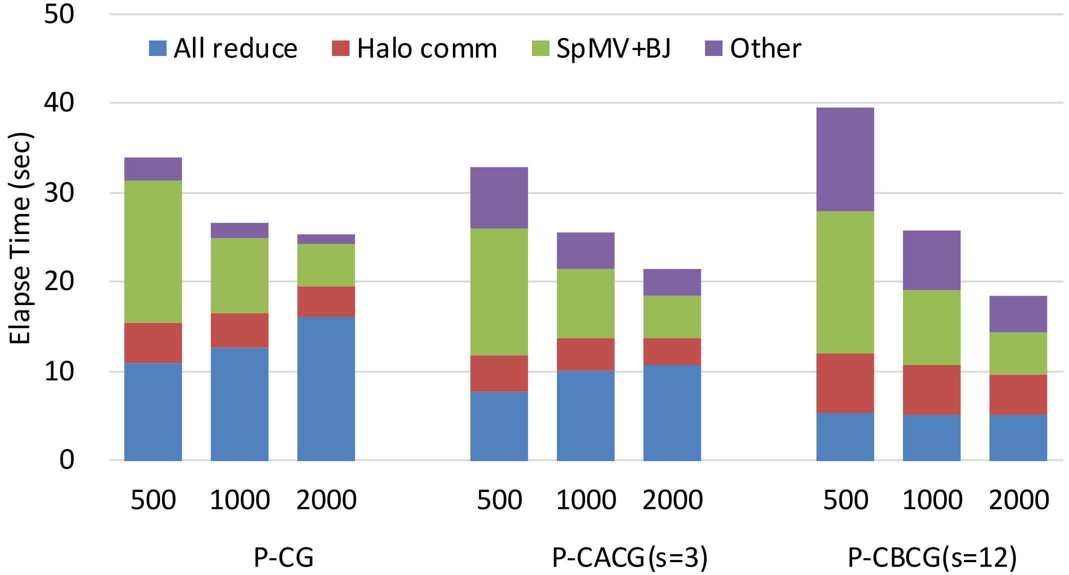
Cost distribution of JUPITER on Oakforest-PACS

[Idomura,LNCS2018]

Strong scaling of JUPITER with P-CG, P-CACG(monomial basis), and P-CBCG

Problem size: $(N_x,N_y,N_z)=(800,500,3450)$





- Chebyshev basis (CBCG) enables larger CA-steps than Monomial basis (CACG)
- Good strong scaling up to 2,000 KNLs (136k cores)
- In P-CG, cost of All_Reduce increases up to 63% of total cost at 2,000 KNLs
- In P-CBCG, cost of All_Reduce is reduced to 32% of P-CG

→At 2,000 KNL, P-CBCG shows 1.4x speedup from P-CG

Re-design GPU preconditioner

- Block Jacobi preconditioner with Incomplete LU factorization
  - Improve convergence by approximate inverse of block sub-matrices
  - Intra-block cannot be parallelized because of data dependency
  - Re-design data blocks for GPU threads

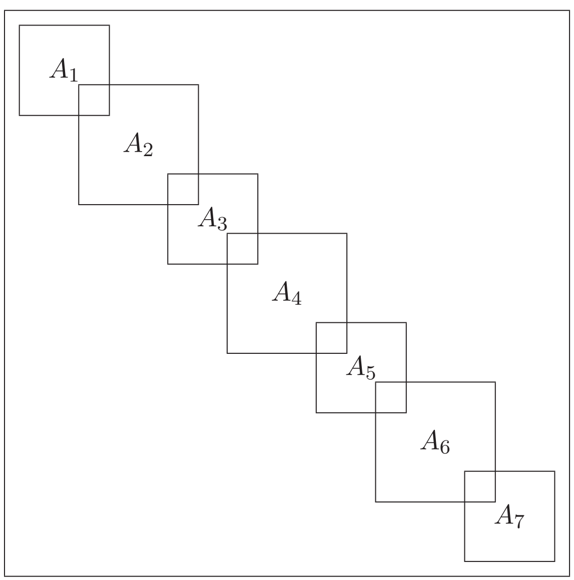
    Data blocks on CPU = 3D domain decomposition (MPI) x ~10 cores

    Data blocks on GPU = 3D domain decomposition (MPI) x ~1,000 cores

    →Convergence degradation due to finer blocks

    →Need to optimize data access patterns on GPU

Block preconditioning



Incomplete LU factorization ILU(0)

For $i = 2, \ldots, n$ *Do:*
For $k = 1, \ldots, i-1$ *and for* $(i,k) \in NZ(A)$ *Do:*
Compute $a_{ik} = a_{ik}/a_{kk}$
For $j = k+1, \ldots, n$ *and for* $(i,j) \in NZ(A)$, *Do:*
Compute $a_{ij} := a_{ij} - a_{ik}a_{kj}$.
EndDo
EndDo
EndDo

GPU optimization of the block Jacobi preconditioner

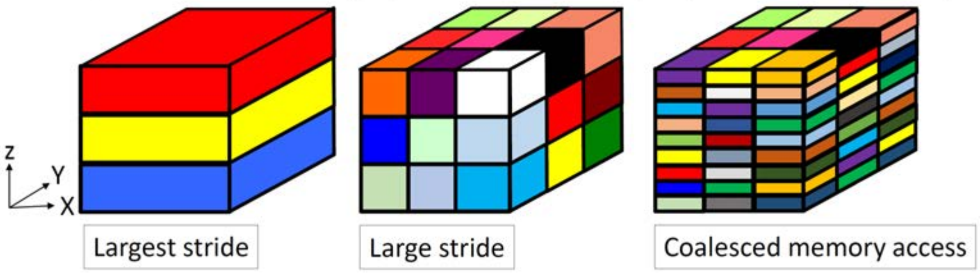
[Ali, GTC Japan 2018]

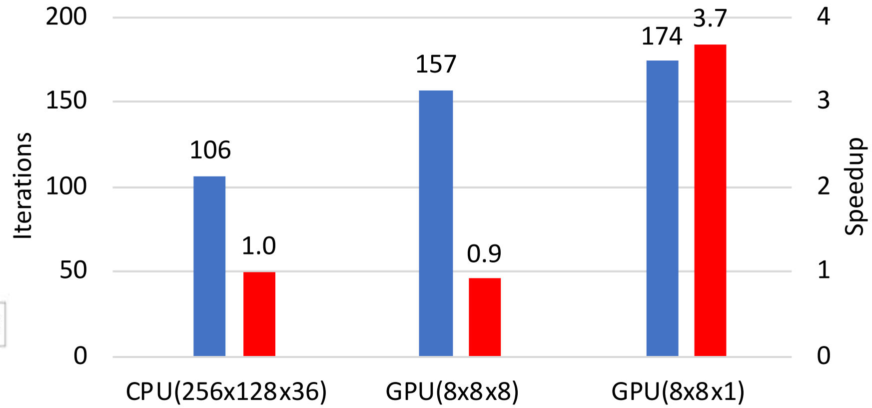
Comparison of P-CG on 1CPU/GPU

Problem size: 256x128x512

- Broadwell (14cores) : 1D block decomposition(z) → 256x128x36~10^6/block
- P100 (3,584cores) : 3D block decomposition(xyz) → 8x8x8=512/block
  - Finer cube blocks lead to 50% increase in number of iterations
  - Slower than CPU because of strided data access
- P100 (3,584cores) : 3D block decomposition(xyz) → 8x8x1=64/block
  - 2D tile blocks lead to 64% increase in number of iterations
  - 3.7x speedup by coalesced data access in z-direction

→Trade off between mathematical and computational properties





Block shape dependency of P-CG solver (JUPITER:256x128x512=1.7M grids)

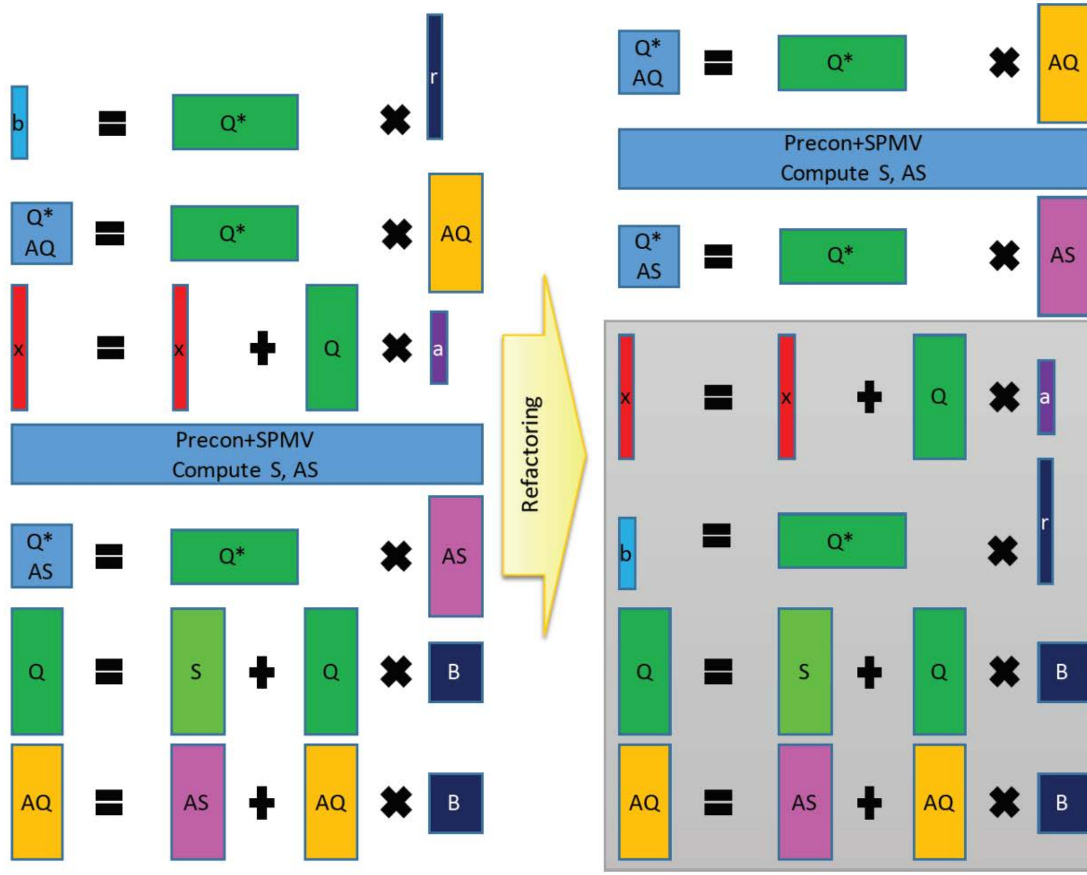
Refactoring GPU kernels

Algorithm Chebyshev Basis CACG (P-CBCG) method

Require: $A\mathbf{x} = \mathbf{b}$, Initial guess $\mathbf{x}_0$

$\mathbf{r}_0 := \mathbf{b} - A\mathbf{x}_0$
Compute S_0 $(T_0(AM^{-1})\mathbf{r}_0, ..., T_{s-1}(AM^{-1})\mathbf{r}_0\ )$
$Q_0 = S_0$
for $k = 0, 1, 2, ...$ until convergence do
  Compute $Q_k^* AQ_k$
  Compute $Q_k^* \mathbf{r}_{sk}$
  $\mathbf{a}_k := (Q_k^* AQ_k)^{-1} Q_k^* \mathbf{r}_{sk}$
  $\mathbf{x}_{s(k+1)} := \mathbf{x}_{sk} + Q_k \mathbf{a}_k$
  $\mathbf{r}_{s(k+1)} := \mathbf{r}_{sk} - AQ_k \mathbf{a}_k$
  Compute
  S_{k+1} $(T_0(AM^{-1})\mathbf{r}_{s(k+1)}, ..., T_{s-1}(AM^{-1})\mathbf{r}_{s(k+1)}\ )$
  Compute $Q_k^* AS_{k+1}$
  $B_k := (Q_k^* AQ_k)^{-1} Q_k^* AS_{k+1}$
  $Q_{k+1} := S_{k+1} - Q_k B_k$
  $AQ_{k+1} := AS_{k+1} + AQ_k B_k$
end for

SpMV+Precon
GEMV
GEMM



- Refactored kernels
  - SpMV
  - Precon
  - Tall-Skinny GEMM (computation for multiple basis vectors)
  - GEMM/GEMV (reuse matrix data to reduce memory access)

  cf. Size of each kernel is limited by registers and shared memory

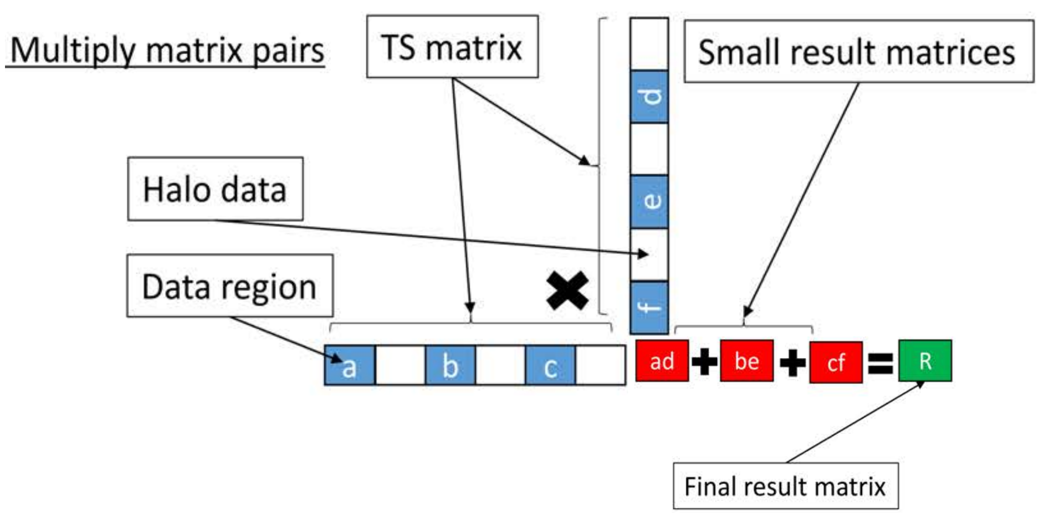
Roofline estimate of CUDA implementation

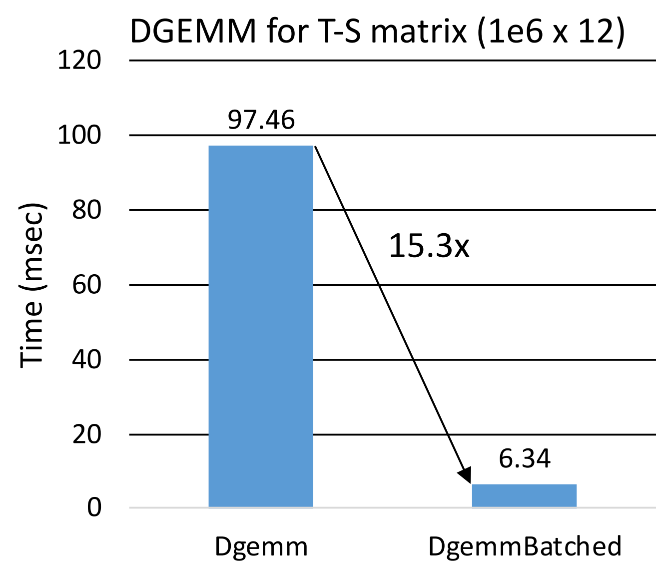
- Roofline estimate for P-CBCG(s=12) CUDA solver on 1 GPU

P100: F=5300GF, B=550GB/s   Problem size: 512x128x256

Kernel	SpMV	Precon	Tall-Skinny GEMM	GEMM/GEMV
Flop/Byte	0.165	0.156	1.108	1.04
Blocks	nx*ny*nz/512	560	Chosen by Batched GEMM in cuBLAS	128
Threads	512	64		288
Roofline time/grid(ns)	0.170	0.237	0.089	0.101
Elapse time/grid(ns)	0.187	0.272	0.096	0.120
Roofline ratio	0.91	0.87	0.93	0.84

- Tall-Skinny GEMM is optimized by batched GEMM in cuBLAS





Overlap halo data communication with computation

- Hybrid CA approach [Mayumi,ScalA'16@SC16]
  - All_Reduce → Comm. avoiding
  - Halo comm. → Comm. overlap

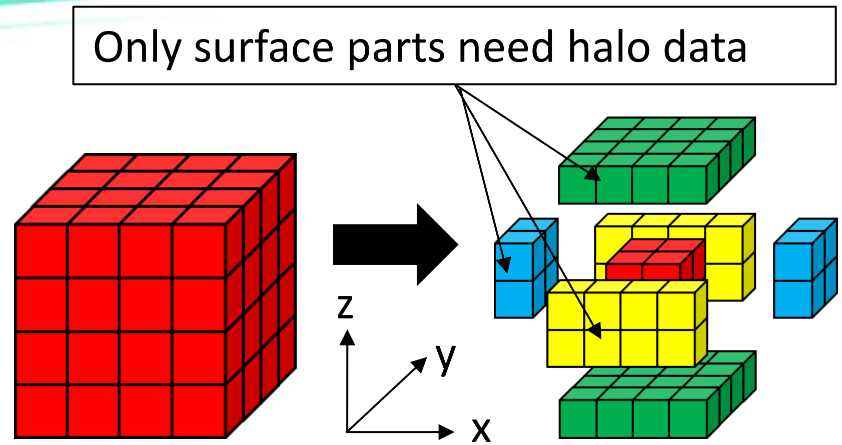
  →Resolve remaining comm. bottleneck in preconditioned CA-Krylov methods

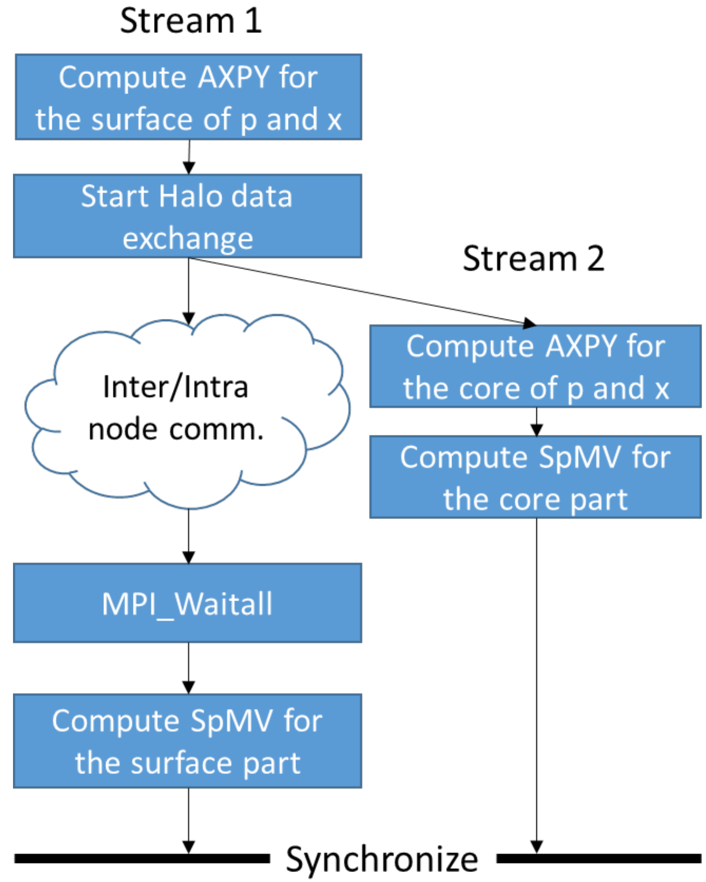
- Divide computing kernels into core and surface parts, and overlap the former
  - Maximize coalesced memory access
  - Overlap multiple CUDA streams

- P-CG provides more overlap
  - P-CG: AXPY→Halo→SpMV

  → 25~30% cost reduction
  - P-CBCG: SpMV→Halo→SpMV

  → 10~15% cost reduction





Strong scaling of P-CBCG on Oakforest-PACS, ABCI and Summit

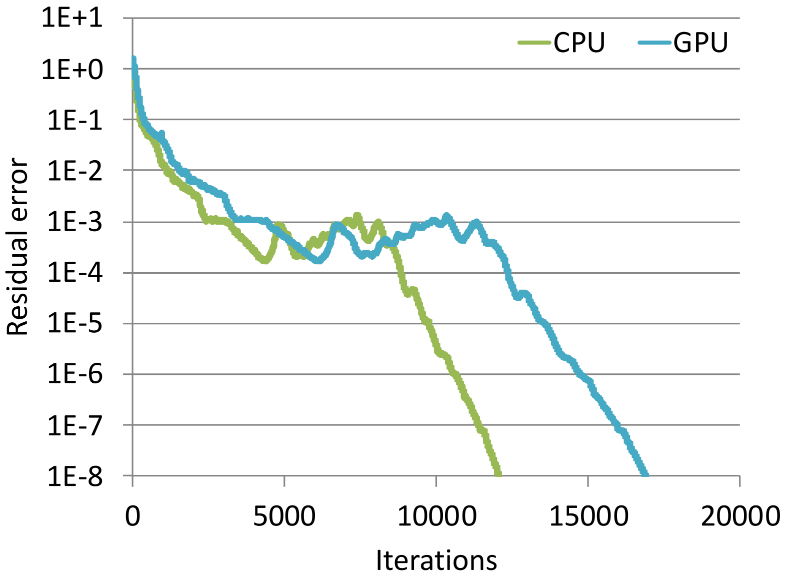
Strong scaling at 512, 1,024, 2,048 KNLs/V100s

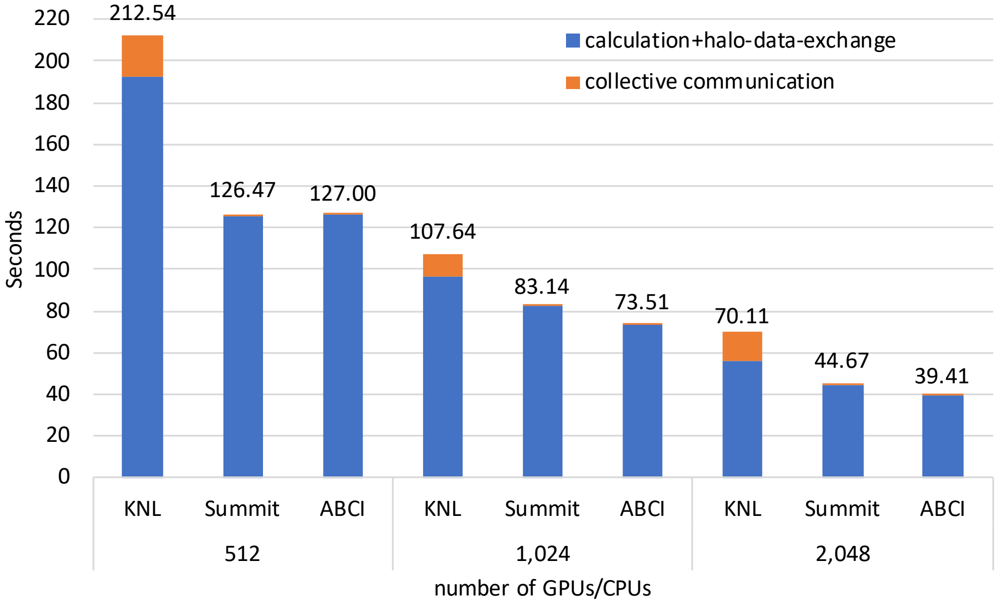
Problem size: 1,280 x 1,280 x 4,608

KNL (Oakforest-PACS): 3.0TF, 480GB/s, Omni-path(12.5GB/s) (1CPU per node)

V100 (Summit): 7.8TF, 900GB/s, IB-EDR4x(25GB/s) (6 GPUs per node)

V100 (ABCI): 7.8TF, 900GB/s, IB-EDR4x(25GB/s) (4 GPUs per node)

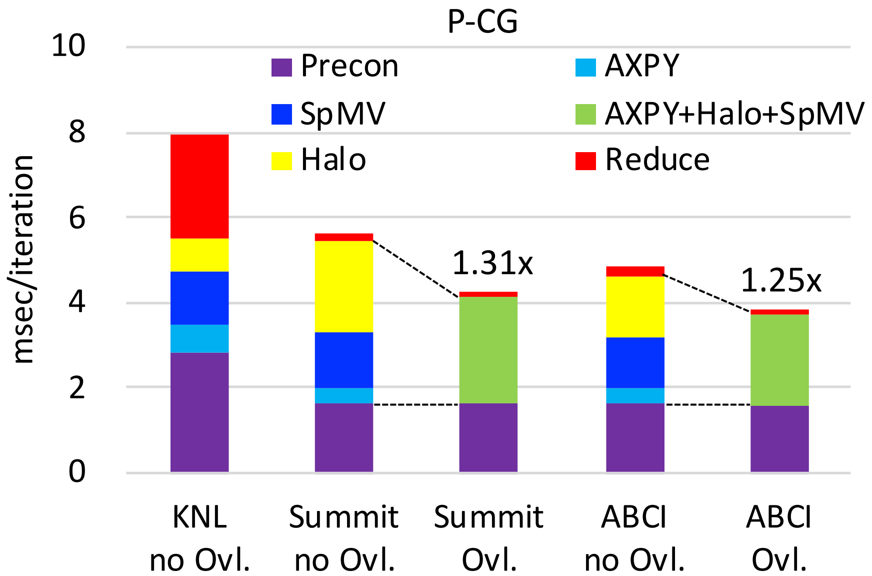


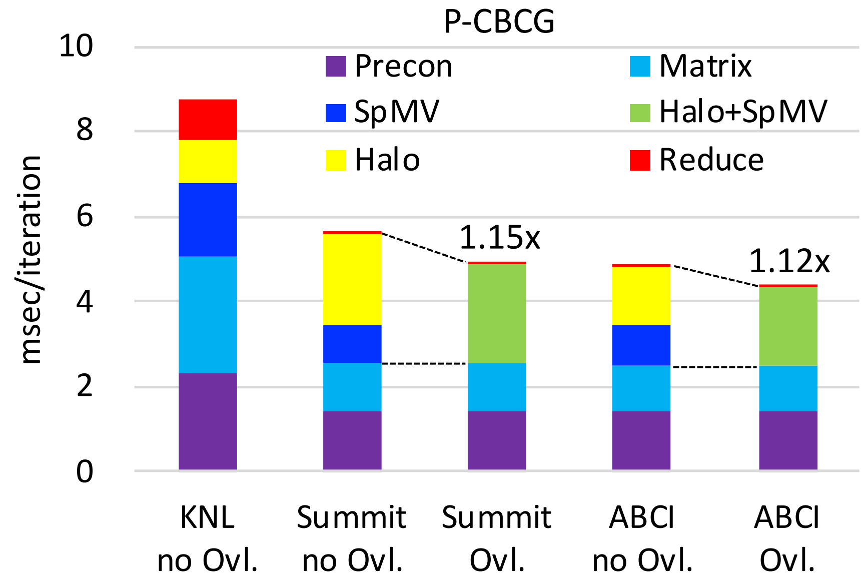


- Block Jacobi preconditioner for GPU requires 1.4x iterations
- ABCI is faster than Summit because of higher interconnect B/W per GPU
- At 2,048GPUs/CPUs, ABCI is 1.8x faster than Oakforest-PACS

Impact of communication avoiding implementation on GPU

Detailed cost distribution at 1,024 KNLs/V100s



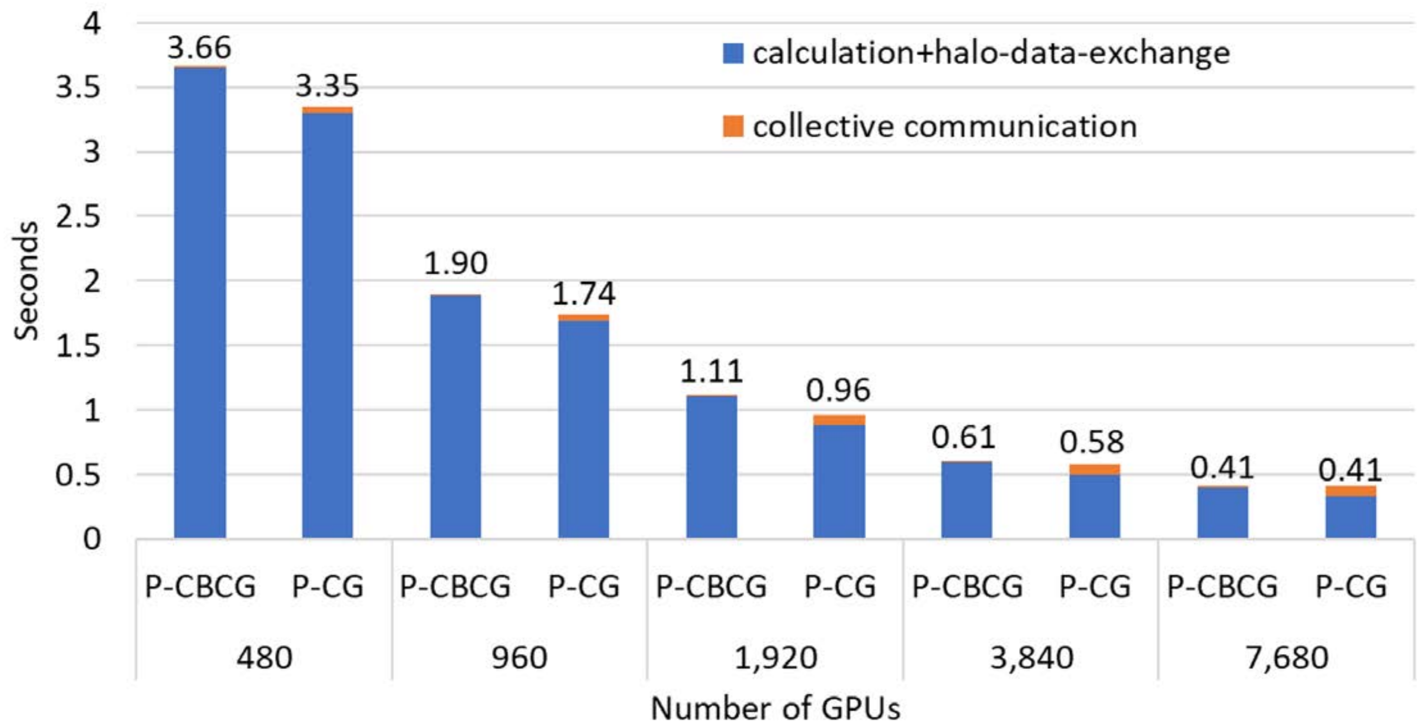


- Computing kernels of P-CG/P-CBCG show 1.5x/2.0x speedups on V100
- All_Reduce on V100 is >10x faster than KNL (flat mode, 64cores x 2SMT)
  →Smaller impact of CA-Krylov methods on V100
- Halo is 2x/3x slower on ABCI/Summit following interconnect B/W per socket
  →Halo data communication is remaining bottleneck on V100
- Communication overlap has significant impact on V100
  →P-CG and P-CBCG are accelerated by 25~30% and 12~15%, respectively

Strong scaling of P-CG and P-CBCG on Summit

Strong scaling at 480 - 7,680 V100s

Problem size: 1,280 x 1,280 x 4,608 (Iterations are fixed to 480 SpMV)



- P-CG outperforms P-CBCG up to 3,840 GPUs

  →P-CG has less computation and larger impact from comm. overlap

- At 7,680 GPUs, both solvers become comparable because of All_Reduce

  →Cost of All_Reduce is reduced from ~20% in P-CG to ~1.3% in P-CBCG

Summary

P-CG and P-CBCG solvers in JUPITER code were ported on ABCI and Summit

- GPU porting
  - Block Jacobi preconditioner was re-designed for >10^3 GPU cores
    - Fully utilized GPU performance, but 1.4x more iterations
  - Refactored GPU kernels achieved 90% of roofline performance
    - Batched GEMM was essential for Tall-Skinny matrix operations
  - Overlap halo data communication and computation

- GPU performance on V100
  - GPU solvers achieved 2x speedup compared with CPU solvers on KNL
  - Bottleneck of halo data comm. was resolved by comm. overlap
  - P-CG/P-CBCG showed good strong scaling up to 7,680 GPUs on Summit
    - P-CG: larger impact from comm. overlap for halo data comm.
    - P-CBCG: less All_Reduce

    →P-CBCG is promising for strong scaling beyond 10^4 GPUs